Microsoft

Operating System

The Windows Installer Service

White Paper

Abstract

This document describes the Microsoft® Windows® operating system Installer service, the needs it addresses, and the methods it uses to address those needs.

The Windows Installer is a new installation service that consists of:

- An operating system-resident installation service
- A standard format for component management
- A management API for applications and tools

Microsoft Corporation • One Microsoft Way • Redmond, WA 98052-6399 • USA

0399

CONTENTS

INTRODUCTION 1

OPERATING SYSTEM RESIDENT INSTALL SERVICE 2

STANDARD FORMAT FOR COMPONENT MANAGEMENT 3
Windows Installer Components 3
Windows Installer Features 4
Windows Installer Products 5
Windows Installer Package File 6

MANAGEMENT API 7
On-Demand Install: Feature Level 7
On-Demand Install: Product Level 8

RUNTIME RESOURCE RESILIENCY 10

OTHER WINDOWS INSTALLER BENEFITS 11
Transacted Installation (Rollback) 11
Source Resiliency 12
Upgrades and Patching 12
Customization Transforms 13
Operation in Lockdown Environments 13
Windows Installer Services within Managed Environments 14

FOR MORE INFORMATION 15
Management and Overview Papers 15
Technical Papers 16

INTRODUCTION

Microsoft developed the Microsoft® Windows® operating system Installer service in response to customer feedback regarding existing installation technologies. Customers wanted an installer that better addressed the needs of corporate deployment, and one that was more consistent and robust than the various types of available installation technologies. Specifically, customers pointed out that the existing installers fail to:

- Adequately manage shared resources
- Consistently enforce the same installation rules
- Provide easy customization
- Help people decide which pieces of an application they need
- Diagnose and repair configuration problems at application run-time

The Windows Installer is a new installation service that consists of:

- An operating system-resident installation service
- A standard format for component management
- A management API for applications and tools

The following sections describe these components.

OPERATING SYSTEM RESIDENT INSTALL SERVICE

The Windows Installer service is an operating system component. It will be included in Windows 2000 and will also be provided as a service pack for the Windows 95, Windows 98, and Windows NT® 4.0 operating systems[1].

In the past, every application provided its own setup executable file or script. Therefore, each application had to ensure that the proper installation rules (such as file versioning rules) were followed. Furthermore, no central reference for installation rules existed because setup was not considered to be a proper part of the development process; few, if any, best practice guidelines were available for developers authoring the setup routines.

Applications quite often did the wrong things at setup time. For instance, many applications installed an older version of a given file over a newer version of that same file. In addition, legacy installers rarely maintained shared Dynamic Link Library (DLL) reference counts. As a result, installation or removal of a given application often broke existing applications on the computer.

With the Windows Installer service, Microsoft has invested significant effort to ensure that all of the proper setup rules are implemented by the operating system. To follow those rules and avoid the problems outlined above, applications merely need to describe themselves in the standard format. This format is known as the *Windows Installer format*. The Windows Installer service will then perform the installation duties on behalf of the applications.

Future versions of the Designed for Microsoft Windows Logo Program will standardize on the Windows Installer for setup.

[1] The Windows Installer service pack for these platforms will be made available. After the Windows Installer service is installed in the Operating System (OS), it can process installation requests from any Windows Installer-enabled applications.

STANDARD FORMAT FOR COMPONENT MANAGEMENT

Whereas existing installers use procedural scripts to deliver a disjointed collection of files, registry keys, and other resources, the Windows Installer service views all applications as three logical building blocks: *components*, *features*, and *products*.

First, a note on terminology: within the scope of this paper, an *installable resource* (*resource* hereafter) is defined as a file, registry key, shortcut, or any other piece that an installer typically delivers to a computer.

Windows Installer Components

A Windows Installer *component* is the smallest and most fundamental of the three logical containers. A component is a collection of files, registry keys, and other resources that are all installed or uninstalled together. When a given component is selected for installation or removal, all of the resources in that component are either installed or removed. Components are the building blocks that are not exposed to the user; only the setup developer needs any knowledge of which components make up a given application.

A given resource can be part of only one component. For example, no two components will share the same file, whether they are part of the same product or parts of different products. To ship a common file, two applications must ship the same common component. Because of this restriction, components are usually small, consisting of a file and other resources that are very tightly coupled to it such as registration information. A component can be said to own its resources.

Keypath

One of the resources within a component can be designated as the *keypath* for that component. Typically, a file is chosen as the keypath, but a registry value can also be a keypath.

The keypath represents the path to a given component. When an application requests a path to a component, the Windows Installer service returns the path to the keypath resource (typically the path to the key *file*).

The Windows Installer service checks for the existence of the keypath when verifying whether a Windows Installer component is properly installed. If the keypath resource is missing for any reason, the Windows Installer service treats that component as broken. For additional details, see *Runtime Resource Resiliency*, later in this document.

Component Code

Another key concept is that a component is globally unique. This means that a single component is guaranteed to always contain the same set of resources regardless of which application ships it. To maintain global uniqueness, each component is assigned a Globally Unique Identifier (GUID) which is known as the *Component Code*. The globally unique Component Code prevents collisions between components that are meant to be distinct.

Unlike other installation technologies, the Windows Installer service does not directly manage files and other resources on the computer. Instead, the Windows Installer service manages applications at the component level—no resource is installed or removed unless the component that owns it is either installed or removed. Setup developers can therefore specify that two resources will never be installed or uninstalled without each other merely by grouping them in the same component.

Furthermore, the Windows Installer components overcome a traditional limitation of the operating system by enabling effective management of resources other than files. Existing installers manage shared files by maintaining a shared reference count (*refcount*) for each shared file in the system registry, and by not removing a given file until that *refcount* reaches zero. However, that scheme does not apply to other resources such as registry keys because there is no mechanism for maintaining a shared *refcoun*t.

Since the Windows Installer service maintains a shared count at the component level rather than at the file level, and because components are atomic units, a proper *refcount* is maintained for all resources. The Windows Installer service does not remove a component unless there are no remaining applications that are depending on that component.

The Windows Installer maintains component *refcounts* in the form of a client list of product codes (rather than integers). This means that the Windows Installer can identify clients of the resource and keep counts synchronized.

The Windows Installer service model for installation and removal is much simpler than the procedural method used by traditional installers. Existing installation technologies lack the notion of components, they use different procedures for installation and removal, and they cannot perform a *refcount* of non-file resources. As a result, they typically leave many resources behind on the computer after an application is uninstalled or removed. The Windows Installer service, by contrast, has a much cleaner model. Because Windows Installer service can accurately track what a given component has installed and when that component can be removed, applications installed using the Windows Installer service can be uninstalled much more cleanly.

Windows Installer Features

The Windows Installer *features* are the granular pieces of an application that a user can choose to install, and typically they represent the functional features of the application itself. When a user chooses *Custom* in a setup program today, the pieces of the application they are able to select for installation correspond roughly to features.

Essentially, features are groupings of components. It is instructive to view a feature as merely a convenient way to select a group of components for installation. In addition, features can also contain other features—this allows applications to be

hierarchically organized. For example, the Microsoft Word feature within Microsoft Office might contain a sub-feature such as Proofing Tools. When a given feature is selected for installation, all of its components are selected for installation.

The Windows Installer service performs all of its management at the component level, thus eliminating the need for a feature to have exclusive ownership of its components. Two features, whether from within the same application or from different applications, can share a given component without affecting the Windows Installer management scheme. Similarly, it is not necessary for a feature to be globally unique; therefore, they do not have globally unique identifiers.

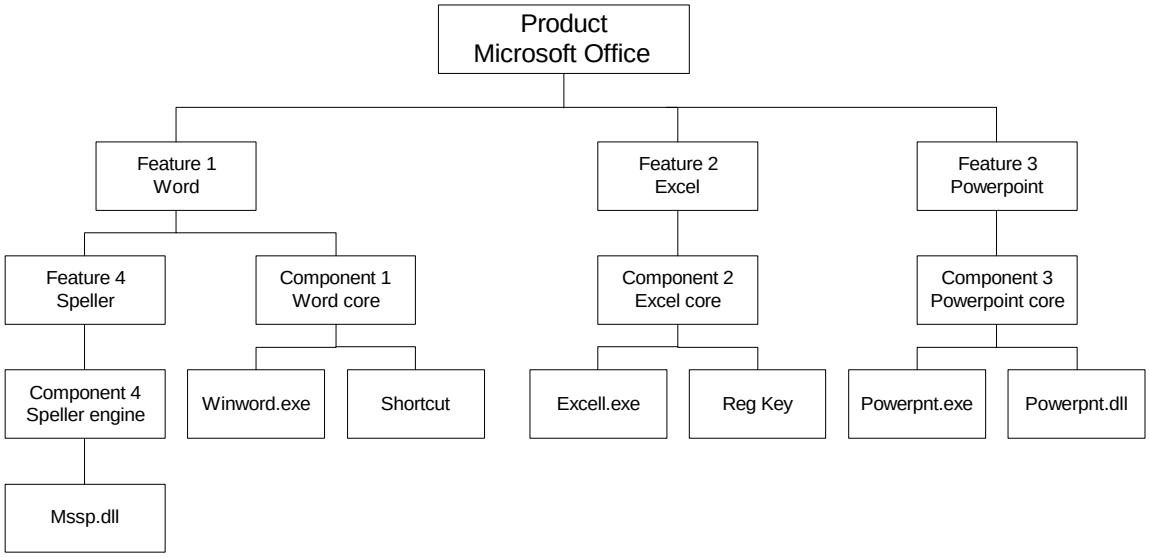
Whereas existing installers typically give the user a binary choice between "installed" and "not installed" for a given feature, the Windows Installer features can be set to one of four states:

- *Installed on Local Hard Disk*—files are copied to the local computer's hard drive.
- *Installed to Run from Source*—files are left on the source (typically a CD or a network share). The application accesses the files from the source.
- *Advertised*—files are left on the source, but they can be automatically copied down the first time they are used. Advertising is explained in the "On Demand Install" section of this document.
- *Not Installed*—no files are copied.

Windows Installer Products

A Windows Installer *product* represents a single product such as Microsoft Office. Products consist of one or more Windows Installer features. Each product is described to the Windows Installer service in the form of a single package file (*.msi file). Products do not directly own any resources, but they do have globally unique identifiers known as *Product Codes*. These Product Codes enable the Windows Installer service to uniquely identify applications that are clients of a given component (the Windows Installer service maintains a list of client products for each component) and to quickly determine if a given product is already installed on a particular computer.

Using a simplified version of Microsoft Office as an example, the following diagram illustrates the relationships between Windows Installer components, features, products, and resources.



Windows Installer Package File

As mentioned previously, each Windows Installer product is described in the form of a single Windows Installer *package file*.

The package file is a database format that has been optimized for installation performance, and describes, among other things, the relationships between features, components, and resources for a given product.

The package file typically appears at the root of the product's CD or network image, alongside the product's files, but it can also contain the product's files in internal compressed cabinet files (.cab).

At installation time, the Windows Installer service opens up the package file for the product in question, and uses the information therein to determine all of the installation operations that must be performed for that product. Those familiar with the previous install technology used by Microsoft Office will note that the package file functionally replaces the .stf, .inf, and .lst files.

MANAGEMENT API

The Windows Installer service provides a management API that enables tools and applications to programmatically:

- Enumerate the products, features, and components installed on the computer
- Install and configure Windows Installer products and features
- Determine the path to specific Windows Installer components installed on the computer

The primary purpose of the management API is to allow the Windows Installer service to manage all file paths on behalf of the application. At run-time, a Windows Installer application can ask the Windows Installer service for a path to a given component. This level of indirection frees applications from a hard-coded dependency on static file paths, which often change from computer to computer or sometimes point to missing files.

Applications that are architected to leverage the Window installer service in this manner gain the advantages of roaming user support, on-demand install, and runtime resource resiliency.

On-Demand Install: Feature Level

With the traditional application installation model, whenever an installation task needed to be performed, a person needed to exit the application and run setup again. The most common such scenario occurred whenever people wanted to use application functionality that they did not install on their first pass through setup. In the traditional model, users needed to anticipate which application features they would require *before* they actually used the application.

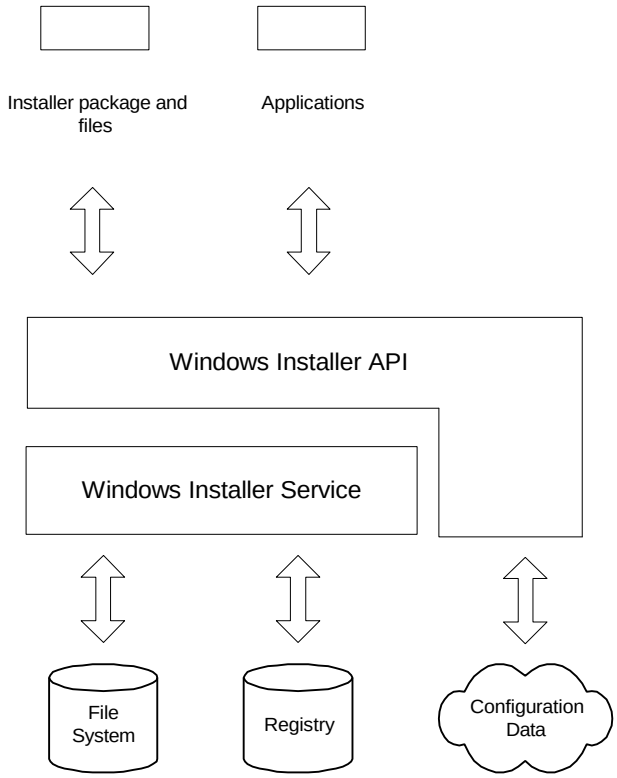
In contrast to the preceding scenario, applications that use the Windows Installer service management API do not require people to run setup again if they decide later that they require additional functionality. If while using such an application a person requests functionality that was not previously installed, the application can call the Windows Installer service to install the necessary feature(s) on the person's behalf; that is, the person does not have to explicitly re-run setup.

In essence, all features of an application are available even if they are not installed. This notion of availability in the absence of the installed files is known as *advertisement*.

For an example of feature-level advertisement, we can look to Microsoft Office 2000, which fully takes advantage of the Windows Installer management API. At setup time, a person can choose to advertise a particular file converter. This file converter is represented by a single Windows Installer feature. If Microsoft Word ever needs that particular converter, it can be installed without running through the original setup. It should be noted that applications can choose whether to prompt the user before performing an on-demand install.

Unlike product-level advertisement, advertisement at the Windows Installer feature level requires no support from the operating system. As such, it is supported on Windows 95 and Windows 98, and on Windows NT 4.0 and later. Product-level advertisement is discussed in the next section.

The following diagram illustrates the basic interaction between an application and the Windows Installer service.



On-Demand Install: Product Level

The Windows Installer service also supports advertisement at the product level. Whereas on-demand install at the feature level requires that the application make use of the Windows Installer management API to install its own features once the application is running, product-level on-demand install requires that the operating system employ the same Windows Installer management API to install advertised applications. Since an advertised application is not installed, it obviously cannot be used to install itself.

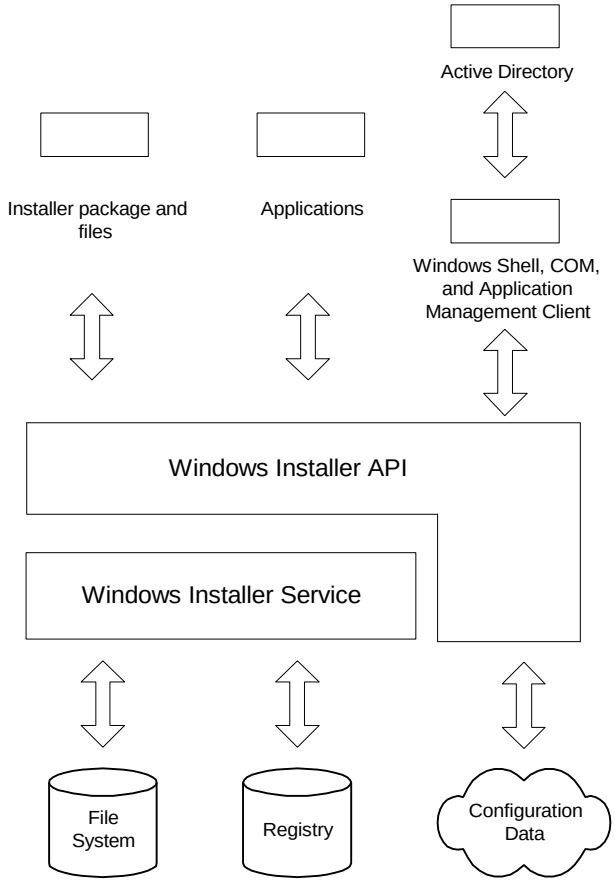
In Windows 2000, both the Windows Shell and OLE make use of the Windows Installer management API. As a result, an entire product can be advertised on Windows 2000. The process of advertising a product merely installs the entry points to that product, such as desktop and **Start** menu shortcuts, file extension associations, and OLE registration.

When a user invokes the activation of the application from one of those entry points,

the operating system calls the Windows Installer service to install the advertised product. When installation is complete, the Windows Installer service returns the path to the newly installed application back to the operating system, which launches the application.

Product-level advertisement is a central piece of the *assign* and *publish*[2] scenarios that Windows 2000 supports with its IntelliMirror™ technology[3].

The following diagram illustrates the interaction between the operating system, the Windows Installer service, the Windows Installer API, and applications.



[2] System administrators can use Software Installation and Maintenance to *assign* applications to people that require the applications. Administrators can also *publish* applications that people may find useful, allowing users to decide whether to install the application. In either case, the application is available without the administrator having to visit each desktop to install it.

[3] IntelliMirror is the attribute of Windows 2000 Change and Configuration Management that combines the advantages of centralized computing with the performance and flexibility of distributed computing. IntelliMirror mirrors people's data and information, applications, and customized users' preferences to a Windows 2000-based server using intelligent caching and centralized synchronization.

RUNTIME RESOURCE RESILIENCY

The Windows Installer management API enables dynamic repair of an application in much the same way that it enables on-demand install. When an application calls the Windows Installer service to resolve a path, Windows Installer service performs two checks.

The first check is to determine whether the requested component and feature are installed. In the case of an advertised feature or product, the component and feature are not installed, and an on-demand installation can be performed.

The second check verifies that all components in the requested feature are properly installed. As mentioned previously, the Windows Installer verifies the existence of the keypath of a given component to determine whether that component is broken. If the keypath resource is missing, an on-demand repair can be performed in the same manner as an on-demand install. An application can therefore repair itself in the course of normal usage.

OTHER WINDOWS INSTALLER BENEFITS

Transacted Installation (Rollback)

When an existing installer encounters a fatal error during setup, the computer is quite often left in an intermediate state; previously installed applications might now be broken, and the new product is not yet fully installed. In these situations, a person is often prevented from doing any work. Also, the person might have neither the time nor the assistance necessary to correct the error. The ability to return the computer to its fully working condition prior to the failed installation would allow the person to continue to be productive using their previous set of applications.

The Windows Installer service provides that ability by maintaining an *undo* operation for every operation that it performs during an installation, removal, or any other configuration change. If something fails during an installation session, Windows Installer service can return the computer to the precise state it was in. This includes restoration of deleted or overwritten files, registry keys, and other resources. Files that are deleted or overwritten during the course of an installation or removal are temporarily saved to a backup location so that they can be restored if necessary. After an installation completes successfully, all temporary backup files are deleted.

As such, an installation cannot be “rolled back” once it has successfully completed; transacted installation is intended as a safety net that protects the computer during a given installation session. If a person wants to remove an installed application, for instance, he or she should simply uninstall that application.

One of the most commonly asked questions regarding transacted installation relates to disk space requirements. As an example, consider an upgrade scenario where an application requires 50 MB of disk space for its files, but can remove 30 MB of files from an older version of that application if it is found on the computer. In this situation, the application will have a net disk space consumption of 20 MB. But since the 30 MB of removed files are temporarily saved, that 30 MB disk space gain is not realized until the installation has completed.

Therefore, the application temporarily requires 50 MB of space to install, but it will still consume only 20 MB of disk space once it is completely installed. It should be noted that *an application will never require more disk space (temporary or otherwise) than the sum total of its installed resources.* In situations where there is enough space for an installation if removed files were not temporarily backed up, but not enough space if they are backed up, the transacted installation can be optionally disabled, allowing the installation to continue.

Setup developers do not need to spend any additional effort in order for their installation to benefit from transacted installation; all Windows Installer service operations natively support rollback on all operating systems.

Source Resiliency

Whenever the Windows Installer service needs access to the source media in order to perform an on-demand installation, reinstallation, or other configuration operation, it has the ability to look for backup sources if the original source is unavailable due to network outage or other temporary problem. At the time of deployment, administrators have the ability to provide the Windows Installer service with a list of backup source locations for that particular product. In addition, if the Windows Installer service cannot find any available source from the list provided, it can prompt the user to browse for a source, and it will then add any 'browsed' sources to the existing list for that product.

Administrators can choose to disable a person's ability to browse for new sources by using Group Policy.

Source resiliency does not require that the application use the Windows Installer management API, and is not dependent on the version of Windows.

Upgrades and Patching

Setup developers can identify related groups of product by defining *Upgrade Codes* (globally unique identifiers) for each group. When combined with a Product Version, an Upgrade Code uniquely identifies a product (much like a Product Code), and can be used to identify older or newer versions of a particular application. In this manner, setup developers can specify whether an older version of an application should be removed, or whether an application should not be installed if a newer version is already installed.

Besides native upgrade support, the Windows Installer service provides built-in patching technology. In much the same way as a Windows Installer package is submitted to the Windows Installer service for installation, a Window installer-based patch can be applied to an existing Windows Installer-installed product in order to upgrade or repair that product. After a Windows Installer patch is applied, it remains on the computer and is used in conjunction with the original source media to provide the bits for on-demand install and resiliency.

In a corporate environment, it is expected that administrators will apply patches to the network installation source point rather than to individual users' computers. In the case where the patch is a Quick Fix Engineering (QFE) patch, the Windows Installer provides both a command line and an API that allows administrators to notify the client workstations that new bits are available.

In the case where the patch actually upgrades a product (as is the case with a full Microsoft Office Service Release), a normal installation needs to be performed from the newly patched network image. The Windows Installer service treats that scenario as a simple upgrade, and as such employs the upgrade logic described above.

Customization Transforms

In the past, if administrators wanted to customize the behavior of an installation, they directly modified the setup script to achieve the desired results. If similar changes were required to many different setup scripts, administrators needed to repeat those efforts for each script.

The Windows Installer service customization transforms modify the Windows Installer package file at installation time, and can therefore dynamically affect installation behavior. Generic modifications can be built into a single transform and applied to many different packages, as long as those modifications are *legal* for the package files in question. For example, a transform that modifies the keypath of a given component is legal provided that the component exists in the package being modified.

Customization transforms, much like patches, remain cached on the computer. These transforms are applied to the base package file whenever the Windows Installer service needs to perform a configuration change (including reinstallation) on the product in question.

Transforms are applied at initial installation, and they cannot be applied to an already installed application.

Operation in Lockdown Environments

To decrease support costs, many organizations have *locked down* their desktops by controlling people's ability to write to the file system and registry. While this prevents a person from accidentally or intentionally modifying their configuration, it also requires administrator intervention whenever a new application needs to be installed.

Since the Windows Installer operates as a system service on Windows NT 4.0 and Windows 2000, it has the ability to run in one of two contexts:

- As the Local System account, which has greater privileges than the user
- As the user, which is the default behavior

In a Windows 2000 environment, using the Group Policy-based[4] Change and Configuration Management, the administrator can approve certain applications, specifying that all configuration operations on those applications (installation, uninstall, and repair) run as the Local System account. In this manner, administrators can lock down the file system and registry as described above, and

[4] Administrators can use Group Policy to define settings for groups of users and computers. These settings include registry settings on the desktop (such as operating system components and applications), scripts (for computer startup and shutdown, and user logon and logoff), software installation options (such as the applications that are available to users and those that appear on their desktop), and security settings (such as for local computer, domain, and network security settings).

the Windows Installer service can still perform installations on the person's behalf. Only those applications approved by the administrator run with elevated privileges.

In a Windows NT 4.0 environment, administrators can specify that *all* Windows Installer transactions run with Local System privileges, but cannot granularly approve certain products.

Windows Installer Services within Managed Environments

The Windows Installer is available on Windows 95 and 98 and Windows NT 4.0, and is built into Windows 2000. It can be used to perform installations locally or it can be used with Systems Management Server 2.0 for centralized control of software distribution and installation.

As mentioned previously, the Windows Installer is an integral part of IntelliMirror, a key component of Windows 2000 policy-based Change and Configuration Management. IntelliMiror leverages Group Policy and the Windows 2000 Active Directory[5] to allow an administrator to assign and publish applications to groups of users or computers within the enterprise. On the client workstation, the Windows Installer service is a core component of the software installation and maintenance feature.

The Windows Installer service provides certain enhanced resiliency features on Windows 2000 desktops, regardless of whether that desktop exists in a Windows Server 2000 managed environment. Product-level advertisement, as previously described, requires operating system support, but that same operating system support also enables resource resiliency when a product is activated using a Windows Installer-enabled shortcut, file association registration, OLE server, or other entry point. These shortcut and file association features require Internet Explorer 4.01 SP 1 shell (or later). This level of automatic installation and repair is therefore also available to other Windows clients, including Windows 98.

[5] Active Directory is a secure, distributed, partitioned, and replicated directory service that provides two key common management services. It provides a standardized location service; that is, it provides a standard way to locate resources within the computer systems. The Active Directory also provides the basis for applying Group Policy to the objects managed by the Active Directory.

FOR MORE INFORMATION

For the latest information on Windows NT Server, check out our World Wide Web site at http://www.microsoft.com/ntserver or the Windows NT Server Forum on the Microsoft Network (GO WORD: MSNTS).

Management and Overview Papers

The following table lists a series of papers that introduce Microsoft's Windows management services and Change and Configuration Management. These papers are intended for managers and technical decision makers who need to understand the business requirements for and the benefits of management features, as well as Microsoft's management architecture, tools, and solutions. We recommend that you read these in the order listed below.

Title	Content	Point your browser to:
Introduction to Windows Management Services	An overview of the management roles and disciplines, as well as the architecture for management solutions that will be available either as part of the operating system or as an add-on.	http://www.microsoft.com /ntserver/management.
Introduction to Change and Configuration Management	An overview of the Change and Configuration Management and an introduction to how Microsoft's products such as Windows 2000 IntelliMirror, Remote OS Install and Systems Management Server address this management discipline.	http://www.microsoft.com /ntserver/management.
IntelliMirror	An overview of the features of Windows 2000 IntelliMirror and scenarios for how organizations can benefit from IntelliMirror.	http://www.microsoft.com /ntserver/management.
Remote OS Installation	An overview of the features of Remote OS Installation and scenarios illustrating how organizations can benefit from IntelliMirror.	http://www.microsoft.com /ntserver/management.
Systems Management Server	An overview of the features of Systems Management Server, and discussion of its benefits.	http://www.microsoft.com /ntserver/management.

Technical Papers

The following table lists additional technical papers that are or will be available for administrators and Information Technology (IT) managers who are interested in understanding the details of Windows management services features and technologies.

More information on	Will be available in this web site:
Active Directory	http://www.microsoft.com/ntserver/management.
Group Policy	http://www.microsoft.com/ntserver/windowsnt5/techdetails/techspecs.
Software Installation and Maintenance	http://www.microsoft.com/ntserver/management.
Remote OS Installation Service	http://www.microsoft.com/ntserver/management.
User Documents and Settings	http://www.microsoft.com/ntserver/management.
Windows Management Instrumentation (WMI)	http://www.microsoft.com/ntserver/management.